Applied Information Technology And Computer Science Vol. 4 No. 2 (2023) 798-811

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Android Battery Saver System

Eastwood Arthur Maclane[1], Suziyanti Marjudi[1*]

[1]Fakulti Sains Komputer dan Teknologi Maklumat,
Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, Johor, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2023.04.02.045
Received 21 June 2023; Accepted 08 November 2023; Available online 30 November 2023

Abstract: An Android Battery Saver System is an application that helps to conserve the battery life of a device. Most smartphones these days have included a built-in battery-saver feature which helps limit their battery consumption. Unfortunately, it is not included in older smartphones or older version of android. With over 83.32% of the world's population having smartphones today, most people are having a hard time maintaining their smartphones' battery life. Therefore, an application which is Android Battery Saver System is proposed. The system is designed using object-oriented approach, and the model used is the agile model as this model is the best for application development. In addition, the application is developed using Android Studio and Firebase to store data in database. Finally, it is expected that Android Battery Saver System can be used optimally and benefit android smartphone users.

Keywords: Battery-saver, Android Battery Saver, Mobile Application Development, Agile Model, Object-Oriented Approach.

1. Introduction

Nowadays, most people use their smartphones to interact with their family, social media, play games, and browse the internet. As a result, using a smartphone must come at a cost because its built-in battery is mostly utilized to keep it powered up and fulfil its portable function. Most smartphones have a battery-saving function built in that helps them use less power. Unfortunately, it is not included on previous Android versions or handsets because most businesses disregarded it. The majority of individuals struggle to preserve their smartphones' battery life despite the fact that over 83.32% of people on the planet own smartphones nowadays. Moreover, most of the battery saver features only provide essential apps that are available for smartphones when it being enabled such as calls, messages, alarm clocks, and calculators. The features forced the user to minimize the user interaction with the smartphones causing their access to their phones to be limited. Older smartphones with older versions of the Android operating system had the most trouble maintaining their battery usage. A solution that assists the majority of Android smartphone users in maintaining and lowering their battery usage as much as possible is thus being suggested. The Android Battery Saver System application was created to eliminate background processes from devices that are consuming excessive amounts of battery power. With the help of the system, users can utilize their use of smartphones entirely without worrying about battery life.

The objective of this project is to develop a mobile application Android Battery Saver System to bring privilege for the user who has older Android versions of smartphone as it aims to help preserve and conserve batteries while utilizing the smartphone fully. The main user of the system will be the

*Corresponding author: suziyanti@uthm.edu.my

customer or the system itself as it will provide a guide or tutorial for the user to explain whatever they need to know on how to use the system. The expected result of the system would be the system can optimize battery consumption, notify the user about their battery status and provide advanced settings for the users to fully utilize their smartphone's battery.

2. Related Work

2.1 Battery Saver Apps and Features

The use of smartphones has been growing massively among people not involving the gender or age of certain people. Santosh (2022) stated that 70 percent of people or even more than that use Android Operating System in the world. Battery saver is a feature available in android devices that aims to make the battery last longer by limiting power consumption. Most Battery Saver features in newer smartphones such as Huawei, Samsung, or Redmi phones have also included Ultra Battery Saving Mode or Ultra Power Saving mode that restricts application usage to only essential applications such as simple calls, SMS, and calculator. The functions of the application and existing features remain the same goal to limit power consumption on smartphones. Figure 2.1 shows an example of Ultra battery Saver Mode UI.

2.2 Battery Saver Method

In most of the Battery Saver features and apps in smartphones, there are several methods that the battery saver can be used to achieve their goals upon extending the battery life. The most common methods that are being used in the feature are the Mode Activation method and the Cleaner & Acceleration method. The Mode Activation method is designed to optimize the performance of smartphone devices by reducing the load on the processor and other system resources. It can be enabled or disabled anytime in the device's system settings by turning it on or off. The mode activation also disables certain features or services such as location services, Bluetooth, Wi-Fi, and Mobile Data. The Cleaner & Acceleration method is focused on improving the overall performance and speed of smartphone devices. This can include clearing the cache, uninstalling unused apps, and disabling unnecessary services. It can also help extend the battery life of the device by reducing the power consumption of unnecessary apps and services.

Table 1: Comparison of the related work

Features/System	DU Battery Saver	GO Battery Saver & Power Widget	Battery Optimizer & Cleaner	Android Battery Saver
One-tap Optimization	√	√	√	X
App Monitoring	X	X	√	√
Battery Monitoring	√	√	√	√
CPU optimization	√	X	X	X
Smart Hibernation	X	X	X	√
Features/System	DU Battery Saver	GO Battery Saver & Power Widget	Battery Optimizer & Cleaner	Android Battery Saver
Junk file & cache cleaner	X	√	√	X

3. Methodology

This section describes all the necessary information about the methodology that is applied to obtain the result of the project

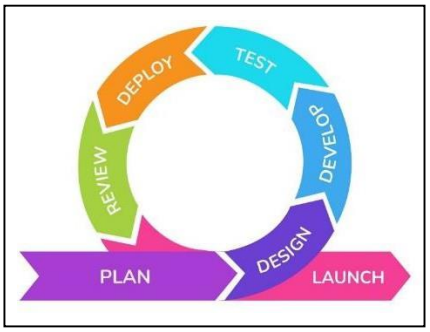

Figure 1: Agile Methodology

3.1 Agile Model Phase

Agile software development and testing follow a process that helps to deliver a working product that provides value at the end of each sprint. Short repetitive cycles are the key to agile software development technique as the approach focuses on involving as many stakeholders as possible. Agile Methodology also helps software projects in raising expectations when the system is ready to determine which items can be safely included in the release cycle. The planning phase is to define the project visions and goals, as well as assign tasks that are needed to implement the system

3.2 Requirement Specification Phase

Requirement specification phase is the second phase for the software development. After the problems and issues that needed to be solved in the project are confirmed, the objectives and the scope of the system project is finalized. This section discusses functional requirements of the system and non-functional requirements of the system for the Android Battery Saver app.

- Functional Requirements System

The functional requirements of this system show an overview of the functions of the modules found in the Android Battery Saver System. Table 3 describes the functional requirements of the system and all the operations that the system is capable of doing in each input, process, and output module.

Table 2: Functional Requirement System Android Battery Saver System

Module	Function
App List Module	Identify and display a list of apps that are consuming the battery power.
App Detail Module	Display information of an app to make changes to a certain app.
Battery Status Module	Provide the user with an estimated time remaining for the current battery charge and suggestion of optimization from the data about the device's battery usage.
Optimization Module	Provide the user to use one-tap optimization to optimize the performance of the smartphone.
Notification Module	Display notification to the user regarding their battery status.
User Consent Module	Prompt the user to allow the application to make changes to the device's system.

Module	Function
Device Information Module	Display the device detail to the user regarding the smartphone that they are currently using.
Advanced Setting Module	Provide user with a guide to enable developer mode.

- Non-Functional Requirements System

The non-functional requirements of the system describe aspects other than the functions of the system that affect the operation of the system. All non-functional requirements of the Android Battery Saver System are described in Table 4.

Table 3: Non-Functional Requirement System Android Battery Saver System

Non-Functional Requirements	
Performance	Reasonable operation and response time of the operating system should be expected.
Compatibility	Application should be able to work on the android platform with version 6.0 and above.
Usability	General appearance and flow of the application are easily understood by all users.
Security	Protect the privacy and security of user's data, including any data collected about app usage or power-saving settings.
Reliability	Reliable and consistently available to the user, with high uptime.

3.3 Design Phase

After the process of gathering requirements and specifications needed to implement the system, the design phase will begin. The Android Battery Saver app will be designed using Unified Modelling Language (UML) which is used to design the system's use-case diagram, class diagram and flowchart for this project. These diagrams are designed so that they can be used in the implementation phase of the project.

- Use Case Diagram

Use-case diagrams illustrate a system's high-level functionality and scope. Use-case diagrams also show how the system and its actor interact with one another. Figure 2 below shows the interaction between the actors and its functionalities for the proposed Android Battery Saver System.

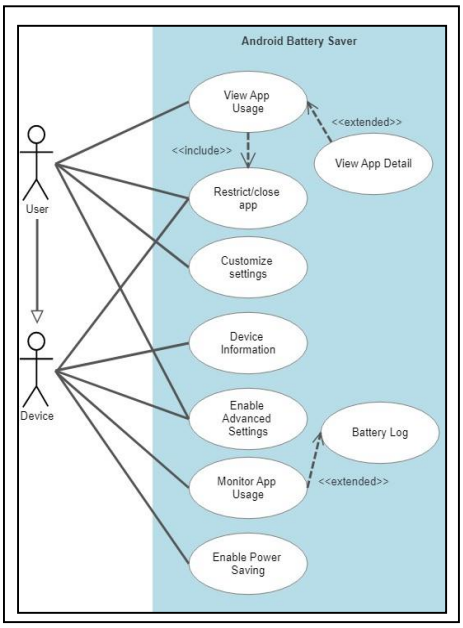

Figure 2: Use Case Diagram

- Class Diagram

The class diagram is used to assist in developing code for the creation of software applications. The class diagram contains the class name, attributes, operations, and the relationship between the class. Figure 3 below shows the class diagram for the proposed Android Battery Saver System.

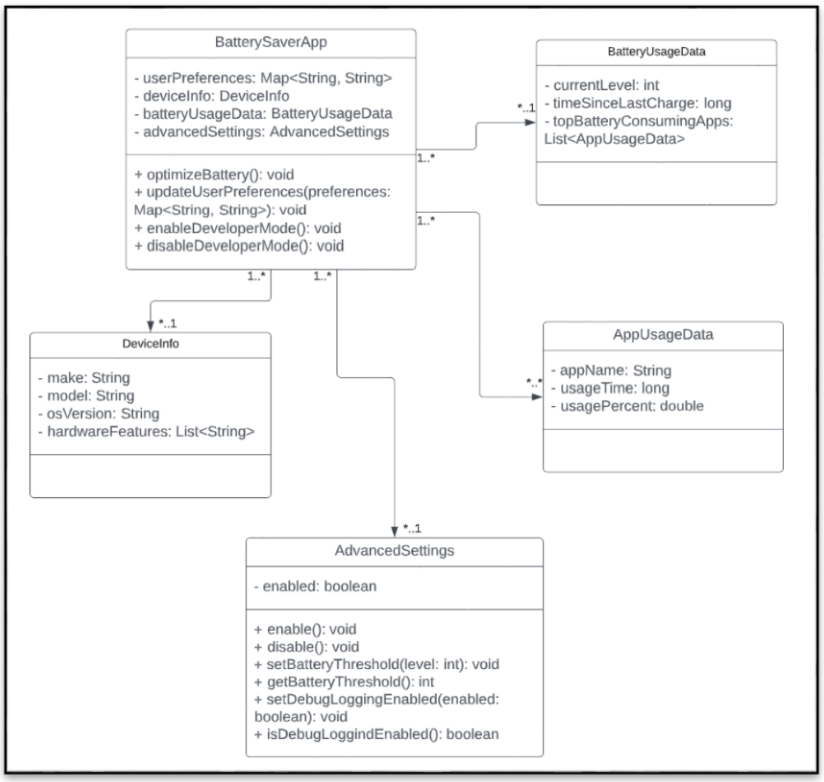

Figure 3: Class Diagram

- Flow Chart

System flow charts are created at the beginning of the design phase to provide an overview of the system process flow that will be produced. Figure 4 below shows the flow chart for the proposed Android Battery Saver System.

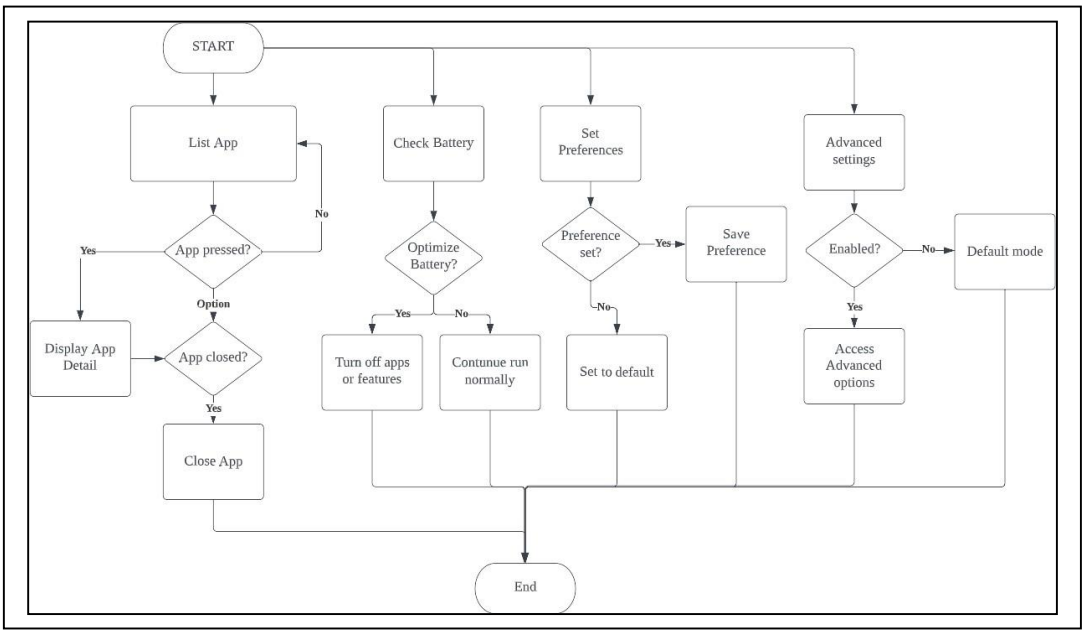

Figure 4: Flow Chart

3.4 Implementation Phase

The implementation phase will begin after the Android Battery Saver app is designed. The requirement gathering and the design that have been made during the analysis and design phase ensure a smooth implementation process. The UML and flowchart design that are produced will be used as a guide in the programming and development process of the Android Battery Saver App. There are six modules that are implemented as functional module development for the system.

- Permission Request Module

The permission request is used to handle necessary permissions that are required for accessing information and implementing optimizations for devices. Figure 5 and 6 show the coding and interface for permission request functional module.

```
Future<void> _checkPermissionsAndFetchApps() async {
  // Check if activity recognition permission is granted
  final permissionStatus = await Permission.activityRecognition.request();
  if (permissionStatus.isGranted) {
    // Permission granted, fetch running apps
    await _getRunningApps();
  } else {
    // Permission denied, handle accordingly (e.g., show an error message)
    print('Activity recognition permission not granted');
  }
}
```

Figure 5: Coding for permission request in dart file

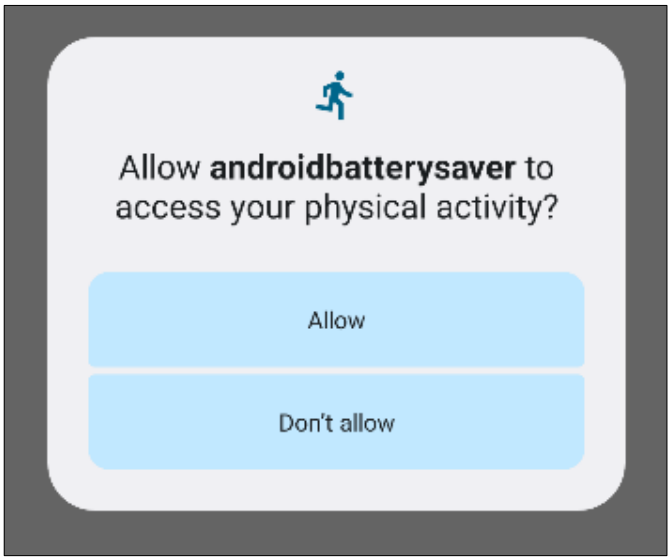

Figure 6: Interface for permission request

- Display Running Apps Module

The display running apps module is used to analyze, detect, and close any applications that are excessively using resources while in the background. Figure 7 and 8 show the coding and interface for displaying running apps.

```
Future<void> _getRunningApps() async {
  try {
    final packageInfo = await PackageInfo.fromPlatform();
    setState(() {
      _runningApps = [packageInfo.packageName];
    });
  } catch (e) {
    print('Error getting list of running apps: $e');
  }
}
```

Figure 7: Coding for getting list of running apps.

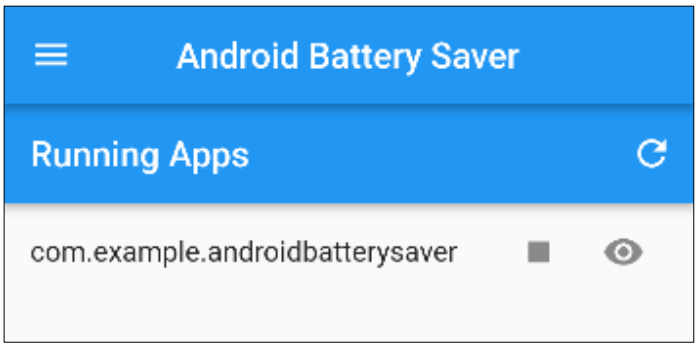

Figure 8: Interface for running apps.

- Battery Status Module

The battery status module is used to monitor battery level, charging status, and other battery-related information. Figure 9 and 10 shows the coding and interface for displaying the battery status.

```
Future<void> _initBatteryState() async {
  try {
    final batteryState = await _battery.batteryState;
    final batteryLevel = await _battery.batteryLevel;
    setState(() {
      _batteryState = batteryState;
      _batteryLevel = batteryLevel;
    });
  } on PlatformException catch (e) {
    print("Failed to get battery status: ${e.message}");
  }
}
```

Figure 9: Coding for displaying battery status.

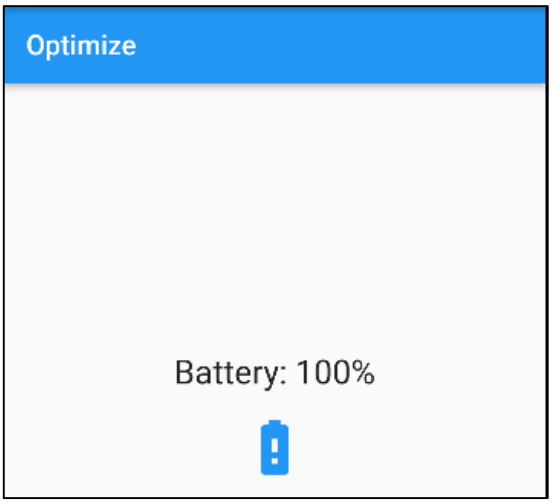

Figure 10: Interface for displaying battery status.

- Optimization Module

The optimization module is used to optimize performance and battery usage of devices by managing cache files and setting the device's brightness to low. Figure 11 and 12 shows the coding for optimizing performance and battery usage.

```dart
Future<void> _optimize() async {
  // Delete cache
  await DefaultCacheManager().emptyCache();

  // Lower screen brightness
  await ScreenBrightness.setScreenBrightness(0.5);

  // Turn on low power mode
  await SystemChannels.platform.invokeMethod<void>('toggleLowPowerMode');
```

Figure 11: Coding for optimizing performance and battery usage.

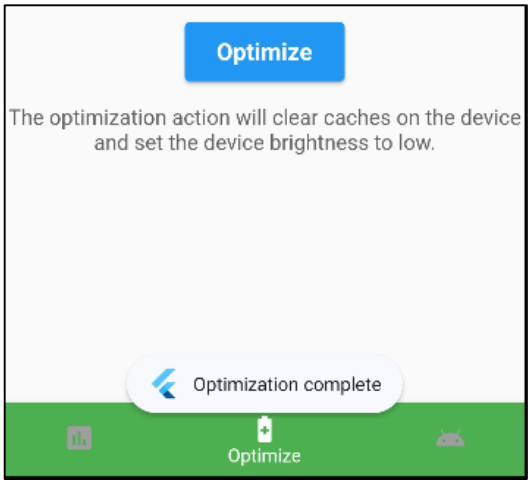

Figure 12: Interface for optimization performance and battery usage.

- Documentation Module

The documentation module is to used provide the user's information and instructions about their devices, and how to use the battery saver module. Figure 13 to Figure 16 show the coding for documentation module.

```dart
Future<void> _getDeviceInfo() async {
  if (await Permission.storage.request().isGranted) {
    try {
      DeviceInfoPlugin deviceInfo = DeviceInfoPlugin();
      if (Theme.of(context).platform == TargetPlatform.android) {
        AndroidDeviceInfo androidInfo = await deviceInfo.androidInfo;
        setState(() {
          _deviceName = androidInfo.device;
          _deviceVersion = androidInfo.version.release;
          _deviceModel = androidInfo.model;
          _deviceID = androidInfo.androidId;
        });
      }
    } catch (e) {
      print('Error getting device info: $e');
    }
  } else {
    print('Storage permission not granted');
  }
}
```

Figure 13: Coding for getting device info.

```
        value: dropdownValue,
        onChanged: (value) {
          setState(() {
            dropdownValue = value!;
          });
        },
        items: [
          DropdownMenuItem(
            value: 'How to enable developer mode',
            child: Text('How to enable developer mode'),
          ), // DropdownMenuItem
          DropdownMenuItem(
            value: 'How to optimize',
            child: Text('How to optimize'),
          ), // DropdownMenuItem
          DropdownMenuItem(
            value: 'How to stop running apps',
            child: Text('How to stop running apps'),
          ), // DropdownMenuItem
        ],
```

Figure 14: Coding for displaying instructions.

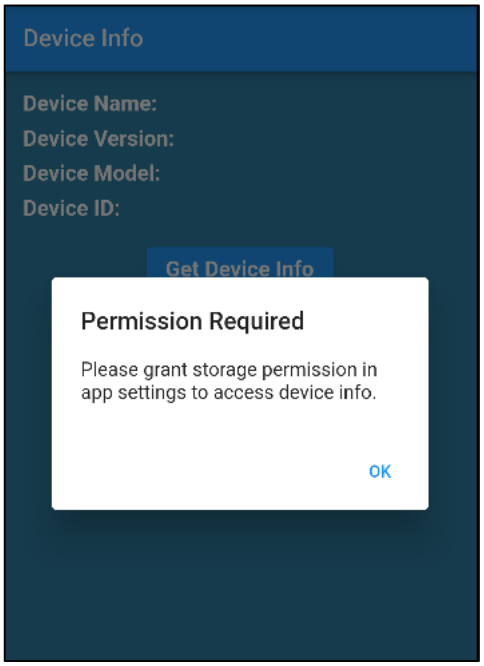

Figure 15: Interface of getting device info (permission denied)

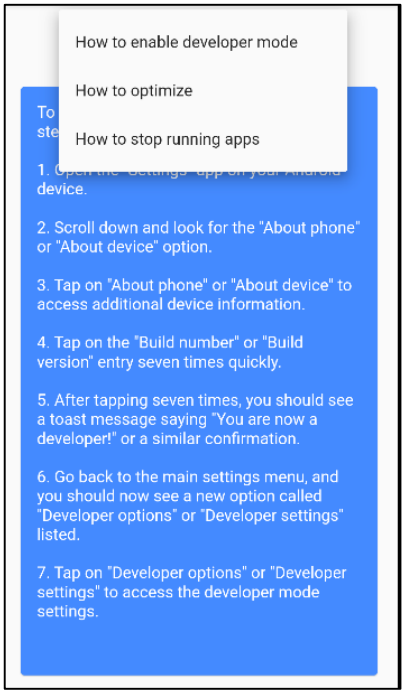

Figure 16: Interface for instructions.

- Advanced Settings Module

The advanced settings module is used to provide users an unessential method to manage their device's resources usage. Figure 17 and 18 show the coding and interface for the advanced settings module.

```
void fetchCpuUsage() async {
  try {
    ProcessResult result = await Process.run('top', ['-n', '1']);
    if (result.exitCode == 0) {
      String? output = result.stdout;
      RegExp cpuUsageRegex = RegExp(r"%Cpu.*?,\s*(\d+\.\d+)");
      Match? match = cpuUsageRegex.firstMatch(output!);
      if (match != null) {
        double usage = double.parse(match.group(1)!);
        setState(() {
          cpuUsage = usage;
        });
      }
    } else {
      print('Error fetching CPU usage: ${result.stderr}');
    }
  } catch (e) {
    print('Error fetching CPU usage: $e');
  }
}
```

Figure 17: Coding for the advanced settings module.

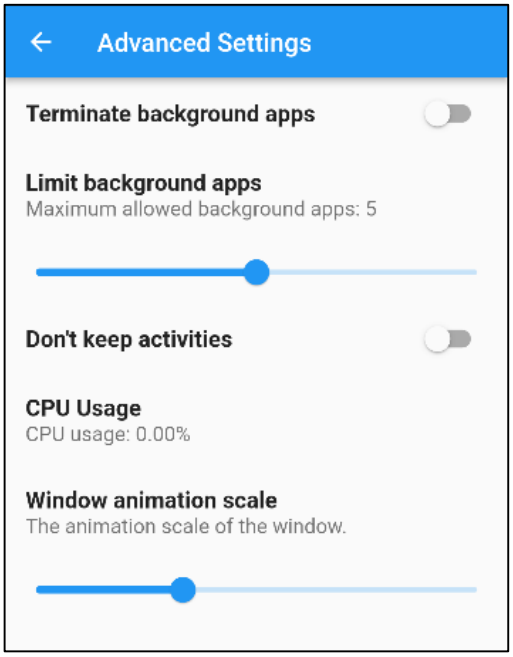

Figure 18: Interface for advanced settings module.

4. Results and Discussion

This section describes the results of the testing that was conducted for Android Battery Saver App on Android for any Android device users. The User Acceptance Test is used to check whether the app has fulfilled all the functional and non-functional requirements. The results of the Functional Testing in the User Acceptance Test are summarized in Table 4.

Table 4: Results of the Functional Testing in the User Acceptance Test

Functional Requirement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1 User can access to every data from the app	0	0	0	0	10
2 User can see all the running apps	0	3	3	4	0
3 The system displays accurate battery percentage	0	1	0	4	5
4 The app clear cache files when optimize button pressed	0	0	2	4	4
5 The app enables low power mode when	4	5	1	0	0

Functional Requirement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
optimize button pressed					
6 User can use advanced settings	2	1	4	3	0
7 The app provides estimation time for the battery usage	0	1	3	1	5

The results of the Non-Functional Checklist in the User Acceptance Test are summarized in Table 3. The results of the Non-Functional Checklist are based on the response given by the test users after they have tested the Android Battery Saver app.

Table 5: Result of the System Checklist in the User Acceptance Test

Non-Functional Requirement	Pass	Fail
1 Optimization have minimal impact on device responsiveness and User experience	8	2
2 Provide error message whenever there is an error occurred	9	1
3 App is compatible with the android devices used	7	3
4 User easy to understand and use the app	8	2
5 User had no trouble on using advanced settings	6	4
6 Provide user guide and instructions for the app	10	0

Based on the results of the User Acceptance Test, it has proven that the Android Battery Saver app has passed the functional testing as most of the users agreed that most of the functional system apps requirements are fulfilled. However, there are several bugs and errors encountered by the user from their feedback as the app itself respond vary with other android devices.

5. Conclusion

The Android Battery Saver System is to provide the older version of Android of smartphone users to converse the battery life of their smartphone. The implementation process can be done when all the modules in the system are working properly and must be done comprehensively. Overall, this mobile application system managed to get the expected results that have been tested and works well to help the users to conserve their battery.

In the future, it is hoped that this system can be added additional functions as most of the features in the newer version of Android system has implemented a better function that could help conserving battery life.

Acknowledgment

The author hereby would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for their support and encouragement throughout the process of conducting this study.

References

[1] Pramanik, P. K. D., Sinhababu, N., Mukherjee, B., Padmanaban, S., Maity, A., Upadhyaya, B. K., ... & Choudhury, P. (2019). Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage. IEEE Access, 7, 182113-182172.

[2] Datta, S. K., & Bonnet, C. (2012, June). Android power management: Current and future trends. ResearchGate.

[3] P. K. D. Pramanik et al., "Power Consumption Analysis, Measurement, Management, and Issues: A State-of-the-Art Review of Smartphone Battery and Energy Usage," in IEEE Access, vol. 7, pp. 182113-182172, 2019, DOI: 10.1109/ACCESS.2019.2958684.

[4] Kang, J., Seo, S., & Hong, J. (2011, September). Usage pattern analysis of smartphones. ResearchGate.

[5] Rusen, C. A. (2022, May). What is Battery Saver (Power Saving)? Turn it on or off on Android devices. Digital Citizen.

[6] Li, D., & Halfond, W. G. (2014, June). An investigation into energy-saving programming practices for android smartphone app development. In Proceedings of the 3rd International Workshop on Green and Sustainable Software (pp. 46-53).

[7] Jeyaranjani, J., Priyadharshini, V., Shailesh, M., & Shreekanth, S. (2022). Android Battery Saving System (No. 7914). EasyChair.

[8] Gomez, A. (2017, March). Can Battery-Saving Apps Really Extend Battery Life? techweez.